배치 프로그램에서 튜닝대상 SQL 추출하기

㈜엑셈 컨설팅본부/DB컨설팅팀 박 성호

"배치 프로그램의 성능문제를 진단하기 위해 트레이스를 사용할 수 없고, 개별 SQL 에 대한 성능점검은 비효율적인 경우에 어떻게 배치 프로그램의 성능문제를 제대로 파악하고 개선안을 도출할 것인가?"

복잡한 로직을 가지고 있는 프로그램 (이후 배치 프로그램)에 대한 성능문제를 파악하기 위해 수행되는 모든 SQL 에 대한 개별 수행내역을 정확히 판단할 수 있어야 한다. 왜냐하면, 특정 배치 프로그램에서 수행되는 모든 SQL 의 I/O 처리량이나 수행시간 등에 대한 정확한 정보를 추출할 수 있다면, 배치 프로그램의 SQL 중 튜닝대상을 선별하는 작업을 효율적으로 수행할 수 있기 때문이다.

따라서 이 문서에서 Oracle 이 제공하는 X$KGLRD 테이블과 SQL 의 수행내역을 조회할 수 있는 Dynamic Performance View 를 활용하여 배치 프로그램에서 수행하는 SQL 의 수행정보를 추출할 수 있는 방법을 제공할 것이다. 그래야만 추출된 SQL 의 수행정보를 정확하게 분석하여 배치 프로그램의 SQL 중 튜닝대상을 제대로 선정할 수 있기 때문이다.

보통 DB 서버에서 수행되는 프로그램 (SQL 이나 배치 프로그램 등)에 대해 성능관리를 위해서, DBMS_APPLICATION_INFO 패키지를 활용하여 MODULE 명을 설정하거나 SQL 에 식별자를 부여하는 방법을 많이 사용한다. 앞으로 이 두 방법을 사용했을 경우에 어떻게 배치 프로그램에서 수행되는 SQL 의 수행정보를 추출할 수 있는지 기술할 것이고, 또한 이 두 방법을 적용하지 않은 경우에 SQL 의 수행정보를 추출하는 방법으로 기술할 내용은 Oracle 의 오브젝트인 PACKAGE/PROCEDURE/FUNCTION 을 이용하여 작성된 배치 프로그램에만 적용되는 점을 미리 알린다.

본격적으로 배치 프로그램의 SQL 에 대한 수행정보를 추출하는 방법에 대해 소개하기 전에 고객 사에서 튜닝요청을 받았던 SQL 목록에 있던 배치 프로그램을 먼저 보도록 하자.

아래의 구문은 튜닝 요청을 받은 SQL 목록에 있던 것 중 하나이다. 그런데 SQL 을 확인해 보니 성능개선이 필요한 SQL 이 아닌, JOB 으로 수행되는 P_POS_TRAN 프로시저에 대한 튜닝요청

을 한 것이었다. P_POS_TRAN 프로시저는 SELECT, INSERT, UPDATE, DELETE 구문을 다양하게 가진 약 5,000 라인의 프로그램으로, 소스 내에 또 다른 프로시저를 호출하는 등 상당히 복잡한 수행 로직을 가지고 있었다. 그런데 이렇게 복잡한 수행 로직을 가지고 있고, 점검해야 할 SQL 의 개수가 많은 프로그램에 대한 성능개선 요청을 받는 경우, 해당 프로그램에서 수행되는 SQL 중 튜닝대상을 선별하는 것은 상당히 어려운 작업이다.

```
DECLARE
        job BINARY_INTEGER := :job ;
        next_date DATE := :mydate ;
        broken BOOLEAN := FALSE ;
BEGIN P_POS_TRAN( 4 , 101 ) ; ---> 오라클 프로시저
    :mydate := next_date ;
    IF broken
        THEN :b := 1 ;
        ELSE :b := 0 ;
    END IF ;
END ;
```

이런 경우 성능개선이 필요한 튜닝대상을 추출하는 가장 효율적인 방법은 트레이스를 통해 수행내역을 분석하는 것이다. 그러나 앞에서 언급된 P_POS_TRAN 프로시저와 같이 데이터에 대한 입력, 변경, 삭제 작업이 있는 배치 프로그램을 운영 DB 서버에서 트레이스를 수행할 수 없다. 또한 개발 DB 서버가 아예 구성되어 있지 않거나, 개발 DB 서버에 프로그램의 테스트에 필요한 데이터가 없다면, 트레이스를 활용하여 튜닝대상을 추출하는 것은 사실상 불가능하다.

이렇게 프로그램에 트레이스를 활용하여 수행결과를 분석하는 것이 적절하지 않는 경우, 프로그램의 성능점검을 위한 또 다른 방법으로 프로그램의 모든 SQL 에 대한 성능점검 (플랜 점검 등)을 수행하게 된다. 그런데 이 방법은 튜닝요청을 받은 후 프로그램에 대한 성능개선안을 도출하는데 까지 많은 시간이 소요될 것이고, 또한 성능문제를 정확하게 판단하지 못할 수 있어 비효율적인 방법이다. 왜냐하면, 프로그램의 전체 수행시간 (Elapsed Time) 중 가장 많은 비중을 차지하는 SQL 을 개선해야 하나, 해당 SQL 이 Loop 구문 안에서 수행되고, 1 회 수행 시 발생하는 I/O 처리량과 수행시간이 타 SQL 에 비해 적어 튜닝대상으로 추출되지 않을 수도 있기 때문이다.

앞에서와 같이 트레이스 수행이나 배치 프로그램의 모든 SQL 에 대한 개별 성능점검을 통해 성능개선안을 도출하는 것이 힘든 경우가 있다. 이럴 때 우리는 배치 프로그램의 튜닝대상 SQL 을

선별하기 위해서 SQL 의 수행정보를 추출하는 방법으로 Oracle 이 제공하는 정보를 원활하게 조회하여 활용할 수 있다면 좀더 쉽고, 효율적이고, 빠르게 배치 프로그램에 대한 성능개선을 할 수 있을 것이다.

그럼 Oracle 이 제공하는 X$KGLRD 테이블과 SQL 의 수행내역을 조회할 수 있는 Dynamic Performance View 를 활용하는 방법을 알아보도록 하자.

테스트를 진행하면서 내용을 확인하기 위해서 먼저 테스트 데이터를 생성하도록 하자.

Script. 배치 프로그램 테스트 데이터 생성

```
* 테이블 생성하기

drop table plsql_t1 purge;

create table plsql_t1
as
select level as c1, chr(65+mod(level,26)) as c2, level+99999 as c3
  from dual
connect by level <= 1000000 ;

* 인덱스 생성 및 통계정보 수집하기

create index plsql_t1_idx_01 on plsql_t1 ( c1 ) ;

exec
dbms_stats.gather_table_stats(ownname=>'exem',tabname=>'plsql_t1',cascade=>true,estima
te_percent=>100) ;

* 프로시저 생성하기

drop procedure plsql_batch_1 ;
drop procedure plsql_batch_2 ;

create or replace procedure plsql_batch_1
as
begin
        delete /*+ BatchTest_plsql_batch_1 */ plsql_t1 ---> SQL 에 식별자 부여
         where c2 = 'aa';
        commit;
```

```
end;
/

create or replace procedure plsql_batch_2
as
begin
      dbms_application_info.set_module('BatchTest',''); ---> Module Name 설정

      insert /*+ BatchTest_plsql_batch_2 */ into plsql_t1 ---> SQL 에 식별자 부여
      select c1, 'a', c3
        from plsql_t1
       where c2 = 'A';
      commit;

      update /*+ BatchTest_plsql_batch_2 */ plsql_t1 ---> SQL 에 식별자 부여
             set c2 = 'aa'
      where c2 = 'a';
      commit;

      plsql_batch_1; ---> 데이터 delete
end;
/
```

배치 프로그램의 수행내역을 확인하기 위해서 앞에서 생성한 PLSQL_BATCH_2 프로시저를 수행한다.

```
SQL> exec plsql_batch_2 ;
```

PLSQL_BATCH_2 의 소스 내용을 보면, DBMS_APPLICATION_INFO.SET_MODULE 으로 MODULE 명을 설정했다. 그리고 INSERT, UPDATE 구문에 SQL 설명을 가지는 주석을 추가하였다. 프로시저에 적용한 이 두 가지는 일반적으로 배치 프로그램이나 단일 SQL 의 성능관리를 위해 사용되는 방법이다. 만약 튜닝요청을 받은 배치 프로그램에 둘 중 한 가지라도 설정되어 있는 경우는 SQL 의 수행정보를 가지고 있는 V$SQLAREA 와 같은 Dictionary View 를 활용하여 튜닝대상을 추출할 수 있다. 그러나 둘 중 어떤 것도 설정되어 있지 않다면 튜닝대상을 추출하는 것은 어려워진다.

그럼 앞에서 언급한 프로그램이나 SQL 에 식별자를 부여한 경우와 부여하지 않은 경우에 따라 어떻게 튜닝대상을 추출할 수 있는지 알아보도록 하자.

MODULE명 또는 SQL에 식별자가 있는 경우

MODULE명이 설정되어 있는 경우

PLSQL_BATCH_2 의 소스 내용을 확인해 보면, 아래와 같이 해당 배치 프로그램에 MODULE 명을 설정하였다.

```
dbms_application_info.set_module('BatchTest',''); ---> Module 명 설정
```

Oracle 11.2.0.3 에서 테스트를 수행한 결과 PLSQL_BATCH_2 프로시저에 적용한 MODULE 명은 PLSQL_BATCH_2 프로시저에서 호출하는 PLSQL_BATCH_1 에도 적용되기 때문에, 배치 프로그램에서 수행되는 모든 SQL 의 수행정보를 V$SQL 의 MODULE 칼럼으로 조회가 가능하다.

해당 배치 프로그램에 MODULE 명이 설정되어 있다고 가정하고, 배치 프로그램에서 수행하는 모든 SQL 중, 총 I/O 처리량이 많이 발생한 순서대로 정렬하여 추출하고자 한다면 아래의 스크립트를 수행하면 된다.

```
select t1.module, t1.substr_sqltext, t1.executions, t1.buffer_gets
from (
       select parsing_schema_name Schema,                                  --> 1
              module,                                                      --> 2
              sql_id,                                                      --> 3
              hash_value,                                                  --> 4
              substr(sql_text,1,37) substr_sqltext                         --> 5
              executions,                                                  --> 6
              buffer_gets,                                                 --> 7
              disk_reads,                                                  --> 8
              rows_processed,                                              --> 9
              round(buffer_gets/executions,1) lio,                         --> 10
              round(elapsed_time/executions/1000000,1) elapsed_sec,        --> 11
              round(cpu_time/executions/1000000,1) cpu_sec                 --> 12
       from v$sqlarea s
      where s.module = 'BatchTest' ---> MODULE 명으로 검색
       order by 7 desc ---> 전체 I/O 처리량이 높은 순으로 정렬
  ) t1
  where rownum <= 50 ;
```

```
MODULE        SUBSTR_SQLTEXT                          EXECUTIONS BUFFER_GETS
------------- --------------------------------------- ---------- -----------
BatchTest     DELETE /*+ BatchTest_plsql_batch_1 */            1      159206
BatchTest     UPDATE /*+ BatchTest_plsql_batch_2 */            1      105014
BatchTest     INSERT /*+ BatchTest_plsql_batch_2 */            1       10901
```

SQL TEXT로 식별할 수 있는 경우

앞에서 PLSQL_BATCH_1, PLSQL_BATCH_2 을 생성할 때, 아래와 같이 개별 SQL 에 식별자를 추가하였다.

```
delete /*+ BatchTest_plsql_batch_1 */ …
insert /*+ BatchTest_plsql_batch_2 */ …
update /*+ BatchTest_plsql_batch_2 */ …
```

배치 프로그램의 모든 SQL 에 식별자를 추가했으므로, 아래와 같이 배치 프로그램의 모든 SQL 에 대한 수행정보를 조회할 수 있다.

```
select t1.module, t1.substr_sqltext, t1.executions, t1.buffer_gets
from (
      select parsing_schema_name Schema,                                --> 1
             module,                                                    --> 2
             sql_id,                                                    --> 3
             hash_value,                                                --> 4
             substr(sql_text,1,37) substr_sqltext,                      --> 5
             executions,                                                --> 6
             buffer_gets,                                               --> 7
             disk_reads,                                                --> 8
             rows_processed,                                            --> 9
             round(buffer_gets/executions,1) lio,                       --> 10
             round(elapsed_time/executions/1000000,1) elapsed_sec,      --> 11
             round(cpu_time/executions/1000000,1) cpu_sec               --> 12
      from v$sqlarea s
     where s.sql_fulltext like '%BatchTest_plsql_batch%' ---> SQL TEXT 로 검색
      order by 7 desc
  ) t1
  where rownum <= 50 ;

MODULE        SUBSTR_SQLTEXT                          EXECUTIONS BUFFER_GETS
------------- --------------------------------------- ---------- -----------
```

```
BatchTest     DELETE /*+ BatchTest_plsql_batch_1 */          1      159206
BatchTest     UPDATE /*+ BatchTest_plsql_batch_2 */          1      105014
BatchTest     INSERT /*+ BatchTest_plsql_batch_2 */          1       10901
```

MODULE명 또는 SQL에 식별자가 없는 경우

앞에서 PLSQL_BATCH_1, PLSQL_BATCH_2 프로시저 생성 시 MODULE 명 설정이나 SQL 에 식별자를 추가하지 않았다면, 어떻게 튜닝대상을 추출할 수 있을까? 이런 경우 Oracle 이 제공하는 X$KGLRD 테이블과 DBA_OBJECTS.OBJECTID 와 V$SQL.PROGRAM_ID 으로 배치 프로그램에서 튜닝대상 SQL 을 추출할 수 있다.

Note. Oracle 버전이 10*g* 이전까지는 X$KGLRD 를 활용하여 추출해야 한다. 왜냐하면, 10*g* 이후 버전에 V$SQL 나 V$SQLAREA 에 PROGRAM_ID 가 추가되었기 때문이다.

X$KGLRD 활용하기

아래에 X$KGLRD 의 칼럼 정보와 테스트 예제를 통해 사용방법을 알아보도록 하자.

x$kglrd 칼럼 구성 - Oracle Version: 11.2.0.3 에서 추출

```
Column Name             DataType
----------------------- ---------------
ADDR                    RAW(4)
INDX                    NUMBER
INST_ID                 NUMBER
KGLHDCDR                RAW(4)
KGLNAOWN                VARCHAR2(64)
KGLNACNM                VARCHAR2(512)       -----> Procedure & Function Name
KGLNACNL                NUMBER
KGLNACHV                NUMBER
KGLHDPDR                RAW(4)
KGLDEPNO                NUMBER
KGLRDHDL                RAW(4)
KGLNADNM                VARCHAR2(512)       -----> SQL Text
KGLNADNL                NUMBER
KGLNADHV                NUMBER                -----> SQL Hash Value
```

```
KGLRDFLG                  NUMBER
```

Oracle 이 제공하는 X$KGLRD 은 SQL 의 수행정보를 담고 있는데, 특정 PROCEDURE 나 FUNCTION 내에서 수행되는 SQL 에 대해 오브젝트명과 함께 확인할 수 있기 때문에, 특정 배치 프로그램에서 수행되는 모든 SQL 을 추출하고자 할 때 유용하다.

Procedure/Function 명으로 조회하기

PROCEDURE 나 FUNCTION 명은 대문자로 입력되어 있으므로 조회 시 유의하자. X$KGLRD 에서 PROCEDURE 나 FUNCTION 명으로 조회할 경우에 PLSQL_BATCH_2 프로시저에서 호출하는 PLSQL_BATCH_1 도 같이 조회해야 전체 SQL 을 추출할 수 있다.

```
col kglnacnm for a15
col kglnadnm for a37
set pagesize 100

select kglnacnm, substr(kglnadnm,1,37) kglnadnm, kglnadhv
 from x$kglrd
where kglnacnm in ('PLSQL_BATCH_2', 'PLSQL_BATCH_1') ;

KGLNACNM        KGLNADNM                                KGLNADHV
--------------- ------------------------------------ ----------
PLSQL_BATCH_2   UPDATE /*+ BatchTest_plsql_batch_2 */ 3943223768
PLSQL_BATCH_2   COMMIT                                255718823
PLSQL_BATCH_2   INSERT /*+ BatchTest_plsql_batch_2 */  111618107
PLSQL_BATCH_1   COMMIT                                255718823
PLSQL_BATCH_1   DELETE /*+ BatchTest_plsql_batch_1 */ 3094796916
```

SQL 을 SQL Text 로 조회하기

SQL Text 로 X$KGLRD 에서 조회할 경우에는 아래와 같이 수행하면 된다.

```
select distinct substr(kglnadnm,1,37) kglnadnm, kglnadhv
 from x$kglrd
where kglnadnm like '%BatchTest_plsql_batch%';

KGLNADNM                                KGLNADHV
```

```
---------------------------------- ----------
UPDATE /*+ BatchTest_plsql_batch_2 */ 3943223768
DELETE /*+ BatchTest_plsql_batch_1 */ 3094796916
INSERT /*+ BatchTest_plsql_batch_2 */  111618107
```

SQL 을 Hash Value 로 조회하기

DB 서버를 모니터링 시 Hash_Value 를 알고 있을 때, 만약 해당 SQL 이 배치 프로그램에서 수행되는 경우에 어떤 프로그램에서 수행된 것인지 찾아야 할 때 아래와 같이 Hash_Value 로 X$KGLRD 에서 조회하면 확인할 수 있다.

```
select distinct substr(kglnadnm,1,37) kglnadnm, kglnadhv
from x$kglrd
where kglnadhv = 3094796916 ;

KGLNADNM                                      KGLNADHV
------------------------------------ ----------
DELETE /*+ BatchTest_plsql_batch_1 */ 3094796916
```

튜닝대상 추출하기

X$KGLRD 을 활용하여 배치 프로그램에서 수행된 SQL 을 추출 후, 아래와 같이 각 SQL 의 수행정보를 분석 후 튜닝대상을 추출하면 된다.

```
select t1.module, t1.substr_sqltext, t1.executions, t1.buffer_gets
from (
         select parsing_schema_name Schema,                                --> 1
                module,                                                    --> 2
                sql_id,                                                    --> 3
                hash_value,                                                --> 4
                substr(sql_text,1,37) substr_sqltext,                      --> 5
                executions,                                                --> 6
                buffer_gets,                                               --> 7
                disk_reads,                                                --> 8
                rows_processed,                                            --> 9
                round(buffer_gets/executions,1) lio,                       --> 10
                round(elapsed_time/executions/1000000,1) elapsed_sec,      --> 11
                round(cpu_time/executions/1000000,1) cpu_sec              --> 12
         from v$sqlarea s
```

```
        where s.hash_value in (3943223768, 3094796916, 111618107)
         order by 7 desc
  ) t1
where rownum <= 50 ;

MODULE        SUBSTR_SQLTEXT                          EXECUTIONS BUFFER_GETS
------------- --------------------------------------- ---------- -----------
BatchTest     DELETE /*+ BatchTest_plsql_batch_1 */            1      159206
BatchTest     UPDATE /*+ BatchTest_plsql_batch_2 */            1      105014
BatchTest     INSERT /*+ BatchTest_plsql_batch_2 */            1       10901
```

DBA_OBJECTS & V$SQLAREA 활용하기

Oracle 버전이 10*g* 이후부터 V$SQL 과 V$SQLAREA 에 PROGRAM_ID 칼럼이 추가되었다. PROGRAM_ID 칼럼은 DBA_OBJECTS 의 OBJECT_ID 칼럼과 연결이 된다. 그러므로 PROCEDURE 나 FUNCTION 으로 작성된 배치 프로그램의 경우, DBA_OBJECTS 와 V$SQL [V$SQLAREA]을 통해 튜닝대상 SQL 을 추출할 수 있다.

먼저 배치 프로그램 명으로 OBJECT_ID 를 추출한다.

```
select object_name, object_id
  from dba_objects
 where object_name IN ('PLSQL_BATCH_1','PLSQL_BATCH_2') ;

OBJECT_NAME           OBJECT_ID
-------------------- ----------
PLSQL_BATCH_1             61738
PLSQL_BATCH_2             61739
```

DBA_OBJECTS 에서 추출된 OBJECT_ID 값으로 V$SQLAREA 의 PROGRAM_ID 와 연결하여 조회하면 아래와 같이 SQL 을 추출할 수 있다.

```
col substr_text for a30
col module for a15

select substr(sql_text,1,30) substr_text, module, program_id
  from v$sqlarea
 where program_id in (61738, 61739) ;
```

```
SUBSTR_TEXT                    MODULE          PROGRAM_ID
------------------------------ --------------- ----------
UPDATE /*+ BatchTest_plsql_bat BatchTest            61739
DELETE /*+ BatchTest_plsql_bat BatchTest            61738
INSERT /*+ BatchTest_plsql_bat BatchTest            61739
```

앞에서 DBA_OBJECTS 와 V$SQLAREA 를 활용하여, 해당 배치 프로그램에서 수행한 모든 SQL 에 대한 수행정보를 아래와 같이 조회할 수 있다. 그리고 조회된 정보를 면밀히 분석하면 배치 프로그램의 SQL 중 튜닝대상을 추출하는 것은 그리 어렵지 않을 것이다.

```
select t1.module, t1.substr_sqltext, t1.executions, t1.buffer_gets
  from (
         select parsing_schema_name Schema,                              --> 1
                module,                                                  --> 2
                sql_id,                                                  --> 3
                hash_value,                                              --> 4
                substr(sql_text,1,37) substr_sqltext,                    --> 5
                executions,                                              --> 6
                buffer_gets,                                             --> 7
                disk_reads,                                              --> 8
                rows_processed,                                          --> 9
                round(buffer_gets/executions,1) lio,                     --> 10
                round(elapsed_time/executions/1000000,1) elapsed_sec,    --> 11
                round(cpu_time/executions/1000000,1) cpu_sec             --> 12
         from v$sqlarea s
         where s.program_id in ( select object_id
                                   from dba_objects
                                  where object_name in ( 'PLSQL_BATCH_1',
                                                         'PLSQL_BATCH_2') )
         order by 7 desc
  ) t1
  where rownum <= 50 ;
```

```
MODULE       SUBSTR_SQLTEXT                          EXECUTIONS BUFFER_GETS
------------ --------------------------------------- ---------- -----------
BatchTest    DELETE /*+ BatchTest_plsql_batch_1 */            1      159206
BatchTest    UPDATE /*+ BatchTest_plsql_batch_2 */            1      105014
BatchTest    INSERT /*+ BatchTest_plsql_batch_2 */            1       10901
```